eip ESCUELA INTERNACIONAL POSGRADOS

Máster en Programación avanzada en Python para Big Data, Hacking y Machine Learning

Hacking y Pentesting con Python

MÓDULO 04

Networking con Python

Programación de red con Python

Aunque existen varias librerías y herramientas para el desarrollo de programas de red, tales como Twisted o Tornado, los elementos básicos de conexión son los sockets.

En python es posible crear un Socket que actué como cliente o como servidor.

Los sockets cliente, se encargan de realizar una conexión contra un host, puerto y protocolo determinado.

Los sockets servidor, se encargan de recibir conexiones por parte de los clientes en un puerto y protocolo determinado.

Python y Scapy – Introducción a la librería.

Scapy es una potente librería escrita y soportada en Python la cual permite la creación, manipulación e inyección de paquetes de un entorno de red.

Permite realizar labores de descubrimiento y enumeración, similar a otras herramientas tan conocidas como Nmap.

Permite la captura de paquetes y su posterior almacenamiento en ficheros PCAP.

Permite la creación de paquetes de forma programática utilizando la API de scapy desde cualquier script.

Python y Scapy – Introducción a la librería.

Con Scapy no es necesario aprender un lenguaje de programación nuevo, solamente es necesario tener conocimientos sobre la sintaxis y estructura de los programas escritos en Python.

Su uso es simple, puede hacerse de forma interactiva (del mismo modo que se hace con el interprete de Python) o directamente desde una rutina completa de código utilizando un fichero.

El funcionamiento básico de Scapy es simple, se crea uno o varios paquetes, se envían a un destino se capturan de respuestas emitidas.

Python y Scapy – Uso básico.

Consta principalmente de una serie de funciones bastante simples pero que requieren de conocimientos sobre redes y protocolos de comunicación tales como TCP, UDP, ICMP, etc. Es posible crear una o varias capas que conformarán el paquete de datos que se enviará al destino.

Los paquetes creados con Scapy pueden ser altamente personalizados utilizando las clases y utilidades disponibles en la API. Para crear un paquete IP, se puede emplear el siguiente comando:
>>> ipPacket = IP()

Con lo anterior se ha creado un paquete IP con sus valores por defecto y dicha estructura se almacena en la variable “ipPacket”.

Es posible manipular cualquier paquete en Scapy accediendo de forma directa a sus campos, asignando valores o utilizando algunas de las funciones disponibles en para tal fin.

```
>>> ipPacket.src="192.168.1.33"
>>> ipPacket.dst="192.168.1.1"
>>> ls(ipPacket)
```

Se ha accedido directamente a los campos "src" y "dst" del paquete IP y posteriormente se le han asignado valores. También se ha utilizado la función "ls" para listar el valor de todos los campos disponibles.

Python y Scapy – Uso básico.

Además de la función “ls” existen otras que permiten conocer el estado de un paquete de datos:

"ls": permite listar el contenido del paquete

"str": permite obtener una representación String del paquete

"hexdump" permite volcar el contenido del paquete en formato hexadecimal

"del" permite restablecer el valor por defecto de alguno de los atributos del paquete.

Existen otras funciones muy interesantes que permiten, entre otras cosas, la manipulación e inyección de paquetes en la red tales como "sniff" y "sr1".

Con Scapy también es posible crear una pila de paquetes, siendo este es el escenario más común. El objetivo de crear cualquier tipo de paquete en Scapy, es el de poder enviarlo a un destino concreto y analizar las respuestas emitidas de forma programática.

Python y Scapy – Uso básico.

El operador "/" permite la creación de capas en un paquete, separando cada capa de forma lógica, tal como se encuentra diseñado el modelo OSI.

```
>>> layer = IP()/TCP()
>>> layer
```

Los conjuntos de paquetes en Scapy pueden ser creados de la siguiente forma:

```
>>>pkt = Ether()/IP(src="192.168.1.33", dst="192.168.1.1")/TCP(sport=22,dport=22)
```

El paquete esta conformado por varios paquetes que respetan la estructura por capas del modelo OSI partiendo de los paquetes que corresponden a las capas más bajas, hasta las más altas.

El paquete IP contiene los campos correspondientes a la dirección IP de origen y destino.

```
>>> ls(layer)
>>> del(layer.ttl)
>>> layer.ttl=128
```

Python y Scapy – Funciones básicas.

El uso de las funciones listadas por el comando "lsc" suele ser bastante simple y en todo caso, muy útiles para diversas labores relacionadas con el análisis de paquetes y entornos de red.

```
>>> sniffed=sniff(filter="tcp port 22")
>>> wireshark(sniffed)
>>> scan = sniff()
>>> wrpcap("/home/adastra/Escritorio/sniffed.cap", scan)
>>> sniff = rdpcap("/pcaps/sniffed.cap")
```

La función “sniff” permite utilizar una interfaz de red que puede ser indicada por parámetro y ejecutar un proceso de captura de paquetes.

La función “wrpcap” permite escribir en un fichero ".pcap" y almacenar en él los paquetes contenidos en un listado de paquetes, los cuales pueden ser el resultado de una captura activa de paquetes con la función “sniff”.

La función "rdpcap" se encarga de leer los contenidos de un fichero "pcap" y almacenar sus contenidos en una variable determinada.

La función “traceroute” se encarga de enseñar la ruta que siguen los paquetes de datos antes de llegar a su correspondiente destino.

Aunque recibe el nombre de la utilidad "traceroute" en sistemas basados en Unix, se trata de una función que no depende de dicha herramienta, por lo tanto no es necesario que se encuentre instalada en el sistema para que la función pueda ejecutarse correctamente.

Python y Scapy – Funciones básicas.

Con Scapy existen varias funciones para el envío de paquetes, tales como send y sendp.

la función send trabaja sobre la capa 3, es decir en la capa de red en el modelo TCP/IP, donde se manejan protocolos tales como ARP, IP, ICMP, etc. Por esta razón con el uso de la función send no es necesario preocuparse por detalles a nivel de la capa de enlace.

La función sendp trabaja sobre la capa 2 es decir a nivel de direccionamiento físico (MAC, LLC, etc.) por este motivo, es necesario ser cuidadoso en seleccionar la interfaz de red y los detalles adecuados.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

decheverri@grupomainjobs.com

LINKEDIN

Daniel Echeverri
https://www.linkedin.com/in/adastra1/

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados